2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

# Distributed Deep Reinforcement Learning based Indoor Visual Navigation

Shih-Hsi Hsu, Shao-Hung Chan, Ping-Tsang Wu, Kun Xiao, Li-Chen Fu, *Fellow, IEEE*

***Abstract*—Recently, as the rise of deep reinforcement learning, it not only can help the robot to convert the complicated environment scene to motor control command directly but also can accomplish the navigation task properly. In this paper, we propose a novel structure, where the objective is to achieve navigation in large-scale indoor complex environment without pre-constructed map. Generally, it requires good understanding of such indoor environment to make complex spatial perception possible, especially when the indoor space consists of many walls and doors which might block the view of robot leading to complex navigation path. By the proposed distributed deep reinforcement learning in different local regions, our method can achieve indoor visual navigation in the aforementioned large-scale environment without extra map information and human instruction. In the experiments, we validate our proposed method by conducting highly promising navigation tasks both in simulation and real environments.**

***Keywords*—** *deep reinforcement learning; visual navigation*

## I. Introduction

For decades, many researchers in robotics field keep trying to introduce robots into domestic environment, and the most essential problem is the navigation. How to efficiently find a path to the goal, accurately localize current position and safely navigate without collision are always challenging and desirable for the mobile robots. Moreover, while navigating in domestic environment, the traversable space is usually narrow and crowded, and there are some situations which make the navigation tasks even harder, such as the blind alley hidden in the large-scale environment. One basic solution is to use visual data to acquire useful information. In comparison with Laser Range Finder and Light Detection And Ranging (LiDAR), visual data lead to better description of objects and thus identify the environment more comprehensively. Traditional methods, though showing good results, require heavy feature engineering and costly pre-constructed maps in order to localize the robot.

Recently, the rise of deep reinforcement learning (DRL) brings a significant advancement to the robotics field. To deal with navigation tasks, it also inspires researchers with a new challenge: indoor navigation without any feature engineering and pre-constructed map. That is, robot learns how to navigate only from the environment and predefined reward function. However, in the traditional DRL algorithm, the more complex and large the indoor environment is, the more difficult and slow the DRL converges. Also, if the task is challenging, the algorithm may consume plenty of time for training.

Shih-Hsi Hsu is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan (e-mail: r04921010@ntu.edu.tw).
Shao-Hung Chan is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan (e-mail:r06921017@ntu.edu.tw).
Ping-Tsang Wu is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan (e-mail:r05921013@ntu.edu.tw).
Kun Xiao is with School of Automation Science and Electrical Engineering, Beihang University, Beijing, China (e-mail:robin_shaun@foxmail.com)
Li-Chen Fu, Director of NTU Center for Artificial Intelligence & Advanced Robotics, Taipei, Taiwan (e-mail:lichen@ntu.edu.tw)

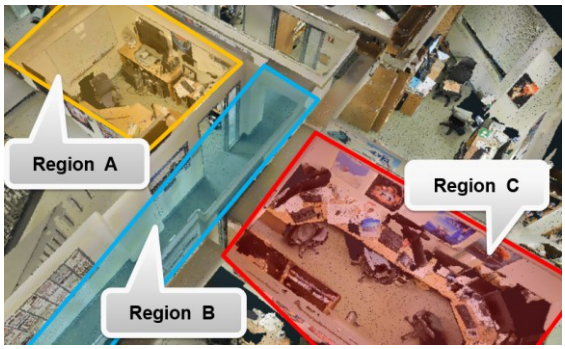



Fig. 1. Our proposed distributed DRL based indoor visual navigation algorithm will train different model in different region in the environment; after training, agent can achieve navigation task from region A to C.

In this paper, we study the problem of DRL based indoor visual navigation and propose a novel learning architecture to enable the robot to find a path leading to the goal in the indoor large-scale complex environment. Due to the high positive correlation between the complexity of navigation task and training time of DRL algorithm, we divide the navigation task to several simpler tasks in different regions, as shown in Fig. 1. The proposed method would train the agents separately in different regions in the indoor environment. Then, the trained models are used to guide the agent to achieve the entire navigation task that is complex and crosses multiple local regions, which are rooms and corridors in the indoor scene. The problem of distributed training is how to know which region model is proper in the environment without knowing the exact current location. Our algorithm can recognize the scene image and decide which region model should be used to achieve the navigation task. The chosen trained local region model would generate the action to lead agent to the goal position. Our method can utilize the distributed training result and achieve better result comparing to traditional DRL method, which would be discussed in the section V.

## II. Related Works

In this section, we first present the current work related to visual navigation, and then discuss the effect of introducing deep learning. Although the works so far have made a great progress on visual navigation, yet the large scale map-less indoor visual navigation still remains a challenge.

### A. Indoor Visual Navigation

Indoor visual navigation is a classical problem that has been studied for long time. The major approaches of visual navigation can be separated into two categories [1], map based method and map-less method. Map based method needs to build the map for the entire environment first and then use it to schedule the rest of task of navigation. Sim et al. [2] presented

an architecture to learn how to automatically select the scene features from landmarks and track them each frame by frame. The tracked features are represented as position and subsequently used in localization task. [3] and [4] proposed an efficient and fast solution for simultaneous localization and mapping (SLAM) using stereo camera and Rao-Blackwellised particle filter. The 3D landmark is extracted and modeled as dynamic Bayes network for indoor localization and grid map contribution.

The other category is map-less based visual navigation that only extract the image feature from the visual sensor and convert it to motor command directly. These methods are efficient to solve the reactive vision based navigation problem [1]. Reactive vision based navigation focuses on how to navigate in the environment and avoid the collision reactively. Talukder and Matties [5] combined optical flow and stereo camera to estimate the depth of scene and they can detect moving objects while robot is also moving. Wang et al. [6] suggested a novel method of constructing paired-landmarks that consist of two characters, which is not only low-cost, high-efficient but also easy to generate more combinations for large scale navigation. Chen et al. [7] proposed a solution of using visual gyroscope for navigation and mapping and unscented Kalman filter is applied for sensor fusion to get accurate robot pose. Although the map-less based visual navigation can use the image feature and directly transfer it to steering control command, the drawback is that global information is not perceived and the path is often not optimal in global view. On the contrary, map-based visual navigation can utilize the map to localize robot's current pose and schedule the path to the goal, yet the map often needs to be constructed beforehand, and the algorithm is often limited to static environment.

### B. *Visual Navigation with Deep Learning*

The combination of deep learning and visual navigation has shown promising result recently. Tai et al. [8] presented a deep network architecture for obstacle avoidance. The raw image is fed into the model to choose proper steering angle. Gupta et al. [9] proposed a deep mapper and hierarchical planner for visual navigation. The mapper network can transform the egocentric view image to top-down free space prediction, and the planner network would use this information to generate next action to the goal. Though deep architecture and achieve good performance, most of them are trained offline and need to collect sample dataset first. The reinforcement learning that learns from reward function instead of labeled data seems to be a feasible solution. The deep reinforcement learning (DRL) that combines deep network and reinforcement learning are also studied in the visual navigation problem. Oh et al. [10] presented a novel memory-based DRL which can conquer the maze environment through many iterations training. The method can deal with the complicated scene image and generate proper actions. However, the target needs to be fixed, which is not able for practical navigation where the target may vary in different contexts. Zhu et al. [11] suggested the target-driven style for DRL model, where the target image is also input for DRL training. Zhang et al. [12] proposed a DRL model whose successor feature is used for recording the characteristic in the previous observation. So far, the deep reinforcement learning has solved many problems in visual navigation field without hand-extracted feature and build-in maps. Nevertheless, visual navigation in large-scale and complex environment is still a difficult task and lack of extensive studies.

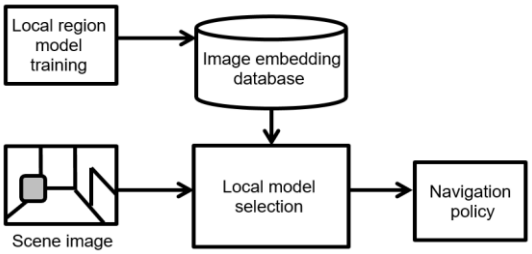



Fig. 2. System overview.

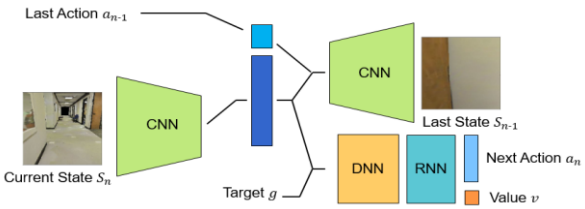



Fig. 3. Proposed local region model.

## III. System Overview

Fig. 2 shows our proposed system overview. Our method is to achieve navigation task in the large-scale complex indoor environment. To relief the growing of task difficulty caused by region area and complexity of path, we divide the whole environment into different local regions and train the agents to achieve navigation tasks in the local regions. After the training results are converged, the scene images are embedded and recorded to the database. This database is used for local model selection. When the agent starts the navigation task, every scene image can be compared to the database and classified to the specific local region. The selected local region model then takes the responsibility for navigation policy and generates navigation action according to the scene image and target position.

## IV. Methodology

In this section, we will describe our proposed architecture. We first introduce our proposed local region model, which is trained with auxiliary task that can speed up the convergent of image embedding vector and stabilize the performance of DRL model. In the second part we will go through the details of the proposed auxiliary task, which is to estimate the last state scene image. In the third part, the whole picture of distributed DRL based visual navigation is shown and discussed.

### A. *Local Region Model*

The model is trained in the local region of the environment, such as rooms or corridors. During every iteration, the convolutional neural network (CNN) takes a state image from the camera and generates the embedding vector, which will be passed to two different networks. One is the de-convolutional network for recovering the vector back to the last-state image. As shown in Fig. 3, the state image $s_n$ is encoded through convolutional network with last action information $a_{n-1}$ and decoded to last state image $s_{n-1}$. This kind of auto-encoder

technique is widely used in many image embedding problem, which can autonomously learn the embedding meaning of image by self-training fashion without supervision. The other network is the reinforcement learning model, which includes fully connected layers presented as "DNN" and recurrent neural layers presented as "RNN in Fig. 3. We use one fully connected layer in the DNN and one long short-term memory (LSTM) [13] layer in the RNN, which provides the ability to memorize past input and take it for consideration to generate next action. The reinforcement learning network would take the learned embedding vector and the target position in the global coordinate into account, and then generates the next action as well as the value of that action, which is similar to actor-critic model [14]. The training skill that combines auto-encoder and reinforcement learning is proposed by [15]. Fig. 4 shows the training process of local region model, where time line counts the time step in iterations of reinforcement learning, and the local region model tries to reach the target point with DRL while reconstructing the last state image from the last action and current state.

### B. *Auxiliary Task*

After introducing our local region model, we discuss the proposed convolutional auto-encoder structure. The network architecture is shown in Fig. 5. The CNN takes current state image, whose width and height are both 256 pixels, as the input and encodes to a 256-dimension vector. Other than the regular auto-encoder [16], with the output of decoder being the same picture as the input, we do slight modifications. First, we change the goal of decoder from recovering the same image to recovering the last state image. Second, we add the last action, which is a 3 dimensional vector $(x_r, y_r, \theta_r)$ in robot's coordinate, as input to the decoder. This strategy can offer the embedding vector causality of action and image pairs, and make deep reinforcement learning training faster and better. Through the proposed auxiliary task structure, the current state image is encoded to the embedding vector in every iteration. The embedding vector, representing the spatial perception, is then used to generate the appropriate action according to the goal.

### C. *Distributed DRL based Visual Navigation*

The distributed DRL based visual navigation utilizes the training results of local region models to have spatial perception in the large-scale environment. Our distributed training contains two steps. First, the indoor environment is separated into several local regions. For example, in Fig. 1, an indoor environment is separated into 3 areas, which are offices and corridor. Then, an agent is assigned and trained in each region. In order to generate the same embedding vector while encoding the same image by different local model, we train our all local models at the same time and every local model shares the same encoder network parameters. This strategy guarantees the latent vector would share the same embedding space, which is important for the distributed training to select the local model later on. On the other hand, the target position is represented in global coordinate, not only in the navigation task but also in the distributed local region model training. Thus, all the local region models can

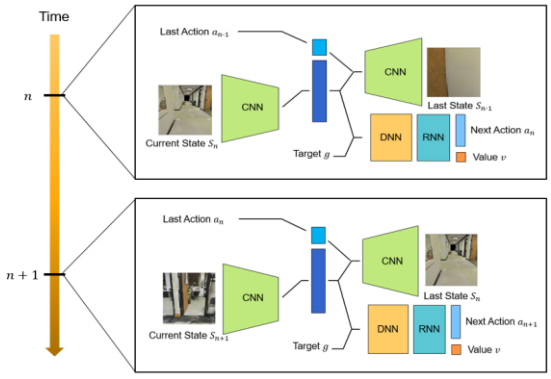



Figure 4. Training process of local region model.

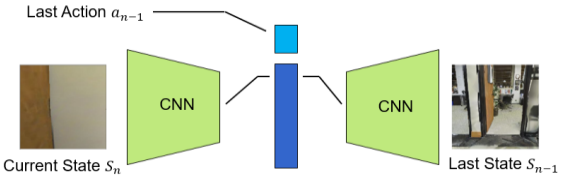



Figure 5. Proposed auxiliary task auto-encoder model.

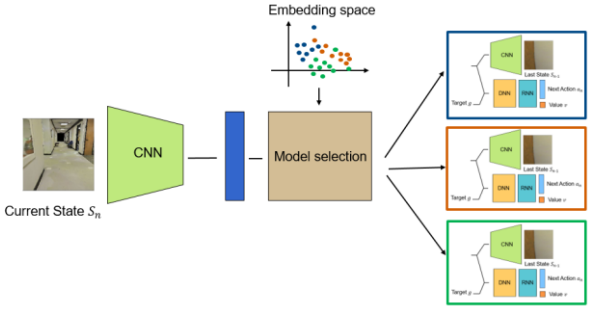



Figure 6. Distributed DRL based visual navigation

share the same coordinate system, this can keep consistence between sharing embedding space and sharing coordinate.

After the training of local region models are converged, the second step is to apply the distributed map-less visual navigation model, which is shown in Fig. 6. The CNN encoder takes the current state image $s_n$ as input and generates the embedding vector. The model selector, which is a classifier trained by the collected state image embedding database, would assign the current vector to a specific local model, generating the action to guide the robot to the target position. In our distributed map-less visual navigation, the model selector finds the most proper local model for each state, while the local model simply focuses on optimizing the navigation performance in the local region. Our method proposed to use plenty of local models to achieve complex large-scale environment navigation, which is extendable and scalable for any size of environment. The training on local model can be efficient and fast since it has less parameters than a huge global model. The selection strategy makes the training experience can share the distributed structure, realize the large-scale complex indoor environment navigation.

## V. EXPERIMENTS

In this section, we will first mention the details of our learning setup, then introduce the experimental results both in

simulator and real indoor environment with real robot.

### A. Learning Setup

The key ingredients of the reinforcement learning setup: action space, observation and goals, reward function design.

*1) Action space:* The action space in our work is defined as discrete actions, such as move forward, turn left, and turn right. We assume that the traversable poses are finite and discrete in the environment. To make training faster, the angel of rotation is fixed to 90 degrees.

*2) Observation and goals:* The observation of the agent is RGB image taken by agent's camera from its egocentric view. The goal is a 3-dimentional vector $(x_r, y_r, \theta_r)$ representing in global coordinate system, which is assumed on the traversable point in the environment.

*3) Reward function:* The decision of reward function affects both the training speed and the final performance. In this work, in order to achieve the efficient navigation in the large-scale indoor environment, we set the reward function as Eq. (1):

$$reward = \begin{cases} 10 & reach\ the\ target \\ -0.01 & otherwise \end{cases} \quad (1)$$

If the agent reaches the target, it will get 10 points; if is not, it will get slight punishment since we would like to increase the time efficiency for navigation task.

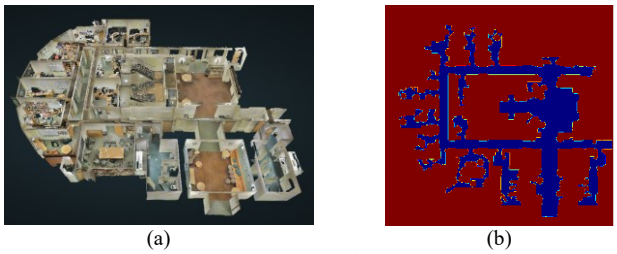
### B. Simulation Experiments

The simulation environment is provided from the open source dataset called Stanford large-scale 3D Indoor Spaces (S3DIS) [17]. The dataset consists of 3D scans, which is in the form of textured mashes, and collected six large-scale indoor environments from 3 different buildings of educational and office use. This large-scale complex indoor data set is suitable and challenging for us to train our method and evaluate it. Although the indoor space is large, the traversable is narrow and limited. For example, Fig. 7(a) shows the indoor environment whose area is 450 $m^2$, but the traversable space which is shown in Fig. 7(b) is relatively narrow. In Fig. 7(b), the blue region is the place where the robot agent can navigate, and the red region is unnavigable. There are many fixed objects in the virtual environment that can be obstacles when agent is executing navigating task. Fig. 8 shows some examples about the objects. In our goal, we want to teach agent to know to recognize and avoid the obstacles by image.

In the simulation environment, the parameters we set are shown in Table I. The height of camera is 120 cm. Every image captured by the camera is 3 channels, 256 x 256 pixel image captured in 60° field-of-view (FOV), which provides us rich information of the environment. The virtual agent in the simulator has volume and collidable, meaning that it can be stopped by the object or the wall in the environment. When the agent is facing a wall but it still wants to move forward, the simulator will warn it and the action is ineffective. We evaluate our algorithm with two different tasks, which is in the different region and different challenges the agent may encounter, as in Fig. 9. TASK 1 starts from a room, goes through a corridor, and moves to another room, where the robot needs to pass the door and avoid from collision when passing corridor. Both two rooms are crowded and full of objects and obstacles, which increases difficulty of task to get out the starting room and reach the goal in another room. By our observation, the regions in TASK 1 can be divided into three parts, which are starting room, corridor, and goal room. We then apply our algorithm on these three regions, as shown in Fig. 10(a). TASK 2 is to move from a hall to another hall. the challenges here are the wide region and long task path, which makes agent need more exploration to reach the goal. We set three local regions in TASK 2, which are starting hall, corridor and goal hall, as shown in Fig. 10(b).

TABLE I. Simulator parameters for virtual agent

| Parameters | Value |
|---|---|
| Camera height | 120 (cm) |
| Camera resolution | 256 × 256 (pixels × pixels) |
| Field of view | 60° (degree) |
| Image channel | 3 (R,G,B) |
| Agent radius | 20 (cm) |
| Agent height | 140 (cm) |
| Step length | 40 (cm) |
| Rotating angle | 90° (degree) |



(a) (b)

Figure 7. (a) Large-scale complex indoor environment simulator.(b) Traversable space in Fig. 7(a)

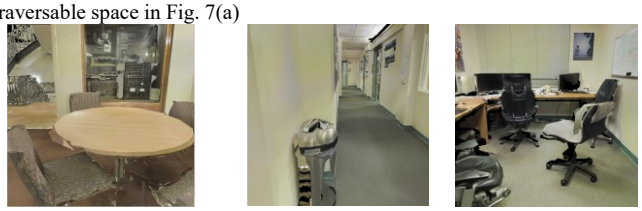
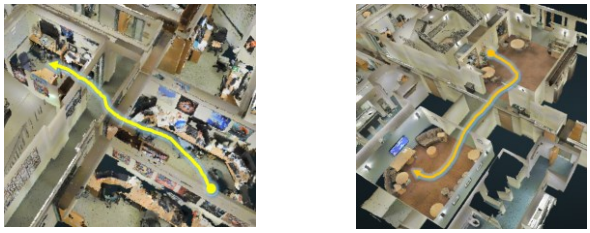
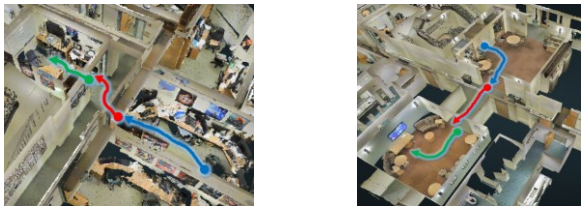


Figure 8. Object examples in the environment, left is a table and chairs, middle is a trach can, right are office chairs.



TASK 1 TASK 2

Figure 9. Different evaluation task in simulator



(a) (b)

Figure 10. Definition of local region in different task

First, we evaluate our proposed model with and without last state image estimation, which is auxiliary task to accelerate the convergence speed. Fig. 11,12 shows the training performance in TASK 1 and TASK 2 trained by proposed model with auxiliary task and without auxiliary task with respect to training episodes the agent takes, which we set to 1200. The orange curves represent our proposed method with auxiliary task, and the green curves represent the method without auxiliary task. Figures from the left column show the length of action sequence agent takes to accomplish the task, while figures from the right column are the total reward of each training episode. Each row of figure is a training result of three different local regions in TASK 1 and 2 respectively.

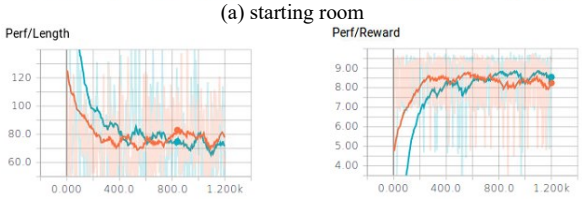
The comparison result shows that our proposed auxiliary task (orange) perform better than the method without auxiliary task (green). This phenomenon not only appears in the beginning of the training, but in the stage for training convergence. Some training curves in certain local regions show that the difference between two curves in the convergence stage are small, even the method without auxiliary task reaches higher value than our proposed method. The reason is that the training curve of the proposed method is about to converge, so the slope of the curves become lower and the learning speed goes slower, and the method without auxiliary task gradually catches up with the proposed method. To sum up, Fig. 11,12 shows that our proposed auxiliary task can help training convergence faster.

After training the local region models, we can evaluate our distributed DRL based navigation. The method we took for comparison is the state-of-the-art in the field of DRL, which is asynchronous advantage actor-critic algorithm (A3C) [14]. The asynchronous training skill makes A3C converge faster and beats other strong competitors. TABLE II. shows the result of comparison. We keep all the conditions of A3C the same as our proposed method except the local region training before task. The indexes we use are the success rate, average hitting times and average length. The success rate is calculated by the mean of 100 testing trails, the success here means the agent starts form the beginning point and reaches the goal point within the time limit, which is 1000 actions; the average hitting times means the average times of agent hitting the wall or obstacle in the success trail; the average length represents the average length of action the agent takes in the success trail. The TABLE II shows that within the same time, which is 12 hours in our experiment setup, our proposed method can achieve higher success rate than the compared method. There is an interesting phenomenon that our success rate is higher, yet the average hitting times and the average length is worse than the compared method. The high success rate and unsatisfying performance give us a future direction that our training in the local regions still can be improved, and the distributed training strategy proved to be effective. The improvement of distributed training is 168 % in the success rate of TASK 1 and 200% in the success rate of TASK 2. The future direction would be to find a better training algorithm in the local region.

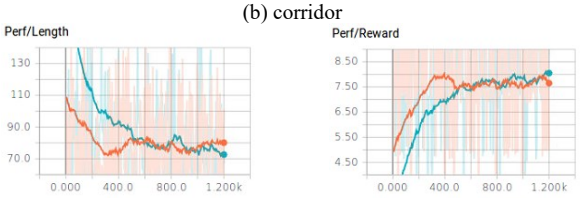
### C. *Real Environment Experiments*

To validate our proposed method in the real indoor environment, we implement our method on the real robot platform. The mobile robot we use is pioneer P3-DX [18], which is a differential wheel mobile platform with two active wheels and one passive steering wheel. To make it close to the height of human, we stack a shelf and place the web camera on it. The camera we use is Logitech web camera providing images with 256 by 256 and 60 Hz. The computation power of the robot is the laptop, ASUS GL552 with NVIDIA Geforce GTX 960M graphic card and i7 6700HQ CPU, as shown in Fig. 13(a). Besides, Fig. 13(b)


(a) starting room

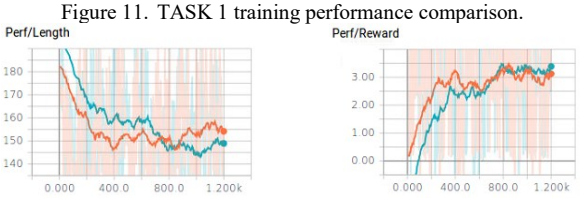



(b) corridor

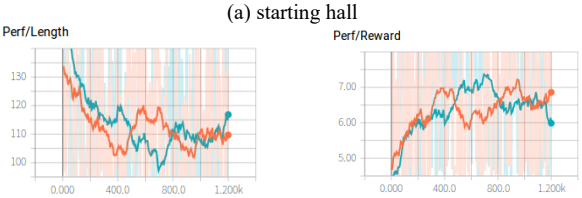



(c) goal room

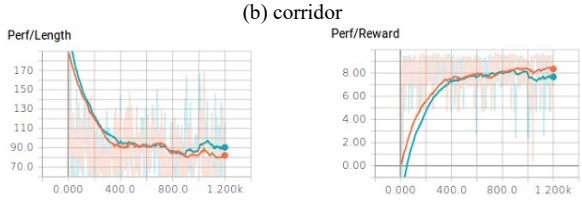
Figure 11. TASK 1 training performance comparison.




(a) starting hall




(b) corridor




(c) goal hall

Figure 12. TASK 2 training performance comparison.

TABLE II. Comparison with different task and final performance

| TASK 1 | Success rate (100/100) | Average Hitting times | Average Length |
|---|---|---|---|
| A3C [24] | 19/100 | 225.32 | 539.07 |
| Ours (proposed) | 51/100 | 436.22 | 689.06 |
| **TASK 2** | **Success rate (100/100)** | **Average Hitting times** | **Average Length** |
| A3C [24] | 16/100 | 142.43 | 837.84 |
| Ours (proposed) | 48/100 | 247.69 | 874.27 |

TABLE III. Final performance of real environment experiment.

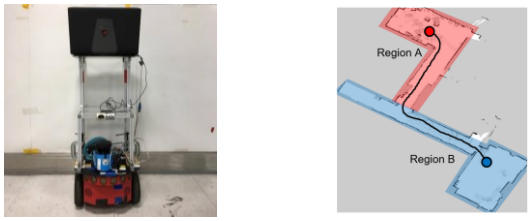
| REAL ENVIRONMENT | Success rate (100/100) | Average Hitting times | Average Length |
|---|---|---|---|
| Ours | 76/100 | 299.03 | 567.53 |

shows the traversable area of the environment. The experiment task is to start form the red point to the blue point. As our observation, we train two different models in two local regions A and B in the experiment, as Fig. 13(b) shown. The step of robot is set to 1m in the experiment and the rotating angle is 90 degrees. Fig. 14 shows the training performance of local region model, the orange curve is the result of region A and the green curve is the result of region B. Left figure shows the sequence of action the robot takes in the local region, and the right figure shows the reward in each episode. The curve converges before 1000 episodes, which means that our method can work in the real environment. We examine the final performance of our method in the real indoor environment. TABLE III is the final result of our experiment. The success rate reaches 76% in this experiment, the average action length is 567.53 and the average hitting times is 299.03. The result shows that our proposed method can be applied on the real robot platform and successfully lead the robot to the target goal.

## VI. Conclusion

In this paper, we proposed a novel deep reinforcement learning (DRL) architecture to solve the map-less indoor navigation in the large-scale environment. We used the auxiliary task to estimate the last state image with the information of the current state image and the last action, which can help DRL converge faster than the conventional one. We also used the target position as an input to DRL, which teaches learning model to transfer the global target coordinate to the policy that leads the agent to the target. The main proposed idea is to use distributed training which recognizes and classifies the current state and asks the corresponding local region model to generate the most optimal action.

The experiment results in simulation showed that the proposed approach is able to make the robot navigate in the complex, large-scale indoor environment and the performance in training is better than the state-of-the-art method. On the other hand, it also reaches 76% success rate in the real environment and proves that our approach is plausible and promising. Due to our assumption, the local region is set by the researchers, which contains a sort of human knowledge inside. In the future, this work can be improved so that one can more intelligently choose the local region model for training. Besides, the current algorithm cannot detect the moving obstacle, and also the human in the environment, which is another possible direction for future works.




(a) (b)

Figure 13. (a) Robot platform. (b) The local region A and B.

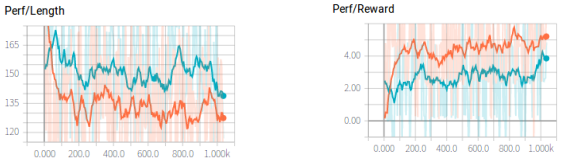



Figure 14. Training performance of real environment experiment.

## VII. Acknowledgment

This research was supported in part by the MOST (Ministry of Science and Technology of Taiwan) Joint Research Center for AI Technology and All Vista Healthcare under the MOST Grant (MOST 107-2634-F-002-018).

## References

[1] Bonin-Font, Francisco, Alberto Ortiz, and Gabriel Oliver. "Visual navigation for Mobile robots: A survey." *Journal of intelligent and robotic systems* 53.3 (2008): 263.

[2] Sim, Robert, and Gregory Dudek. "Effective exploration strategies for the construction of visual maps." Intelligent Robots and Systems, (2003).

[3] Sim, Robert, Pantelis Elinas, Matt Griffin, Alex Shyr and James J. Little. "Design and analysis of a framework for real-time vision-based SLAM using Rao-Blackwellised particle filters." Computer and Robot Vision, (2006).

[4] Sim, Robert, and James J. Little. "Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters." Intelligent Robots and Systems, (2006).

[5] Talukder, Ashit, and Larry Matthies. "Real-time detection of moving objects from moving vehicles using dense stereo and optical flow." Intelligent Robots and Systems, (2004).

[6] Wang, Tianmiao, Yicheng Zhang, Chaolei Wang, Jianhong Liang, Han Gao, Miao Liu, Qinpu Guan, and Anqi Sun. "Indoor visual navigation system based on paired-landmark for small UAVs." Robotics and Biomimetics, (2014).

[7] Cheng, Chen, Wennan Chai, and Hubert Roth. "A single frame depth visual gyroscope and its integration for robot navigation and mapping in structured indoor environments." Journal of Intelligent & Robotic Systems 80.3-4 (2015): 365-374.

[8] Lei, Tai, Shaohua Li, and Ming Liu. "A deep-network solution towards model-less obstacle avoidance." Intelligent Robots and Systems, (2016).

[9] Gupta, Saurabh, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. "Cognitive mapping and planning for visual navigation." arXiv preprint arXiv:1702.03920 (2017).

[10] Oh, Junhyuk, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. "Control of memory, active perception, and action in minecraft." International Conference on Machine Learning. (2016).

[11] Zhu, Yuke, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali, Farhadi. "Target-driven visual navigation in indoor scenes using deep reinforcement learning." Robotics and Automation (ICRA), (2017).

[12] Zhang, Jingwei, Jost T. Springenberg, Joschka Boedecker, and Wolfram Burgard. "Deep Reinforcement Learning with Successor Features for Navigation across Similar Environments." arXiv preprint arXiv:1612.05533 (2016).

[13] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[14] Mnih, Volodymyr, Adrià P. Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning." International Conference on Machine Learning. (2016).

[15] Jaderberg, Max, Volodymyr Mnih, Wojciech M. Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. "Reinforcement learning with unsupervised auxiliary tasks." arXiv preprint arXiv:1611.05397 (2016).

[16] Masci, Jonathan, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. "Stacked convolutional auto-encoders for hierarchical feature extraction." Artificial Neural Networks and Machine Learning–ICANN 2011 (2011): 52-59.

[17] Armeni, Iro, Ozan Sener,Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. "3d semantic parsing of large-scale indoor spaces." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016).

[18] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx